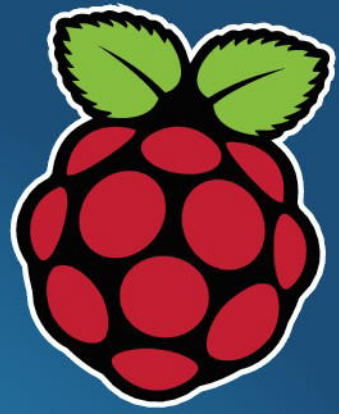
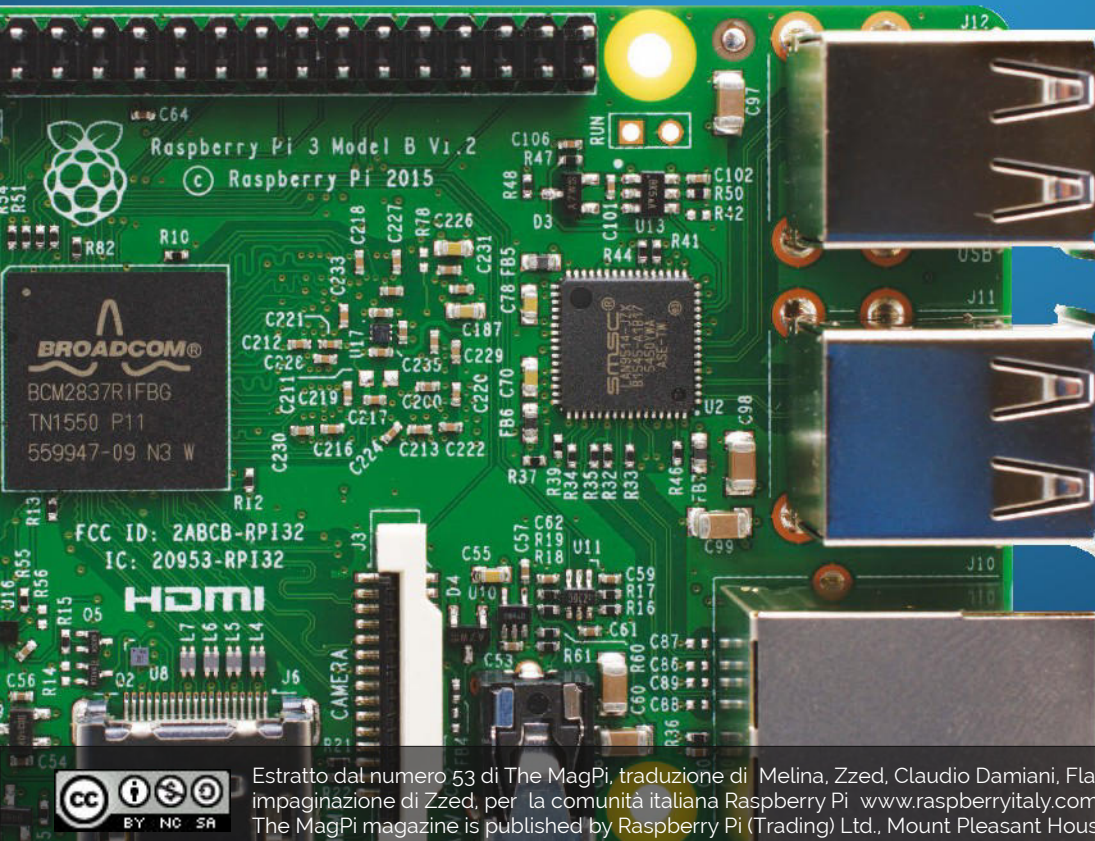
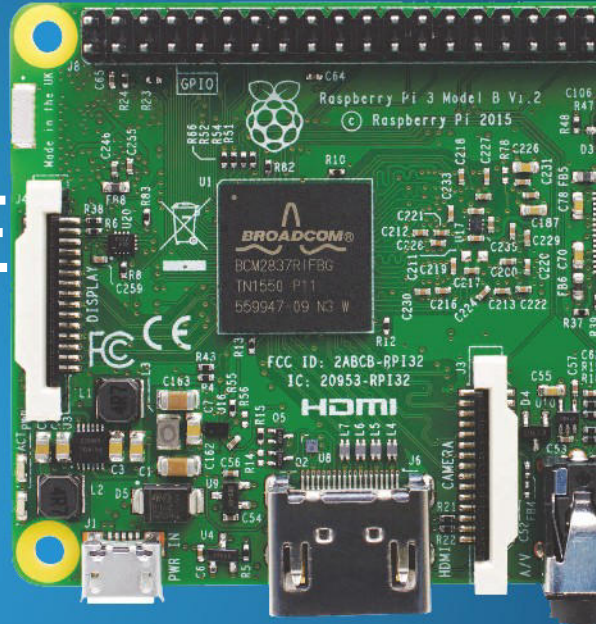
LA TUA RIVISTA RASPBERRY PI **UFFICIALE**

La rivista ufficiale Raspberry Pi in italiano, da RaspberryItaly.com

Numero 53 Gennaio 2017 www.raspberryitaly.com

GUIDA ALLA PROGRAMMAZIONE PER PRINCIPIANTI

Scopri il divertimento di programmare con la più grande e migliore guida introduttiva per il Raspberry Pi

LANCIA PIXEL SUL TUO COMPUTER

Il desktop PIXEL è ora disponibile per computer Intel. Installa Debian con PIXEL sul tuo PC o Mac e usalo proprio come fosse un Raspberry Pi

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Mount Pleasant House, Cambridge, CB3 0RN. ISSN: 2051-9982

L'UNICA RIVISTA PI SCRITTA DALLA COMUNITÀ RASPBERRY PI

Programmazione: GUIDA PER PRINCIPIANTI

Scopri il piacere e l'arte di programmare i computer con Raspberry Pi

Imparare a programmare è una di quelle cose che puoi fare e che può profondamente cambiare la tua vita. Questa è sempre stata una realtà, ma imparare a programmare sta acquisendo sempre più importanza nel mondo moderno.

Il motivo per cui il Raspberry Pi è stato creato è per contrastare il calo delle iscrizioni a Scienze delle Applicazioni Informatiche presso l'Università di Cambridge. I computer moderni e sopratutto le console da gioco, erano divertenti e potenti, ma non facilmente programmabili.

La comunità dei maker si innamorò di Raspberry Pi, grazie alla sua economicità ed alla sua facilità di modifica e hackerabilità. Realizzare progetti, creare e armeggiare con l'elettronica sono i principali motivi per cui noi amiamo Raspberry Pi. Progetti meravigliosi utilizzano una combinazione di hardware e software combinati tra loro.

Quindi, sia che tu sia un hacker che sta imparando a sviluppare miglioramenti per i suoi progetti, o un programmatore alla ricerca di una migliore carriera, quanto segue ha lo scopo di aiutarti nei tuoi intenti.

La buona notizia è che non hai bisogno di essere un genio per saper programmare, proprio come non devi essere un genio per saper leggere e scrivere. Anzi è abbastanza semplice una volta che impari alcuni facili concetti come variabili, salti condizionali e cicli. Forse sei completamente nuovo nel mondo della programmazione. Può anche essere che tu abbia studiato un poco di BASIC a scuola, o tu abbia usato vecchi linguaggi di programmazione tipo Pascal e Fortran. Oppure può essere che tu sia già in grado di sviluppare un progetto e voglia solo imparare il linguaggio che lo controlla.

Qualsiasi sia la tua situazione, siamo qui per accompagnarti attraverso i concetti base della programmazione dei computer. Demistificheremo l'intero processo della programmazione così che tu possa avere una migliore comprensione di quanto succede all'interno del tuo Raspberry Pi.

Questioni di codice

"Penso che chiunque in questo paese dovrebbe imparare a programmare un computer,"disse il cofondatore di Apple Steve Jobs, "perché insegna a pensare."

La programmazione è un qualche cosa di critico che si trova tra le nostre vite ed il mondo sempre più digitalizzato che ci circonda. E' sufficiente una piccola quantità di informazioni su come funziona la programmazione, e sarai in grado di effettuare operazioni al computer in modo più veloce ed avere una migliore comprensione del mondo che ti circonda.
Sempre più, persone e computer lavorano assieme.

Imparare la programmazione e l' utilizzo dell'hardware è incredibilmente motivante. I computer sono veramente vicini all'uomo; si tratta di un aiuto utilizzando la tecnologia. Che si tratti dell'oftalmoscopio fatto in casa che salva la vista in India, o del supporto che il computer consente nel portare internet nelle zone rurali dell'Africa, la programmazione sul Raspberry Pi sta facendo davvero una grande differenza.

La programmazione ti rende inoltre più creativo. Permette infatti di automatizzare una serie di attività della tua vita che sono ripetitive ed estremamente noiose, consentendoti di liberarti di esse e concentrarti su argomenti più interessanti.

Ti insegna inoltre a risolvere alcuni problemi nella tua vita. Imparando a mettere in ordine le varie cose, a spezzettare un grosso problema in una serie di problemi minori, azioni che sembrerebbero impossibili in una serie di azioni minori ma risolvibili, consente di cambiare la propria vita.

E se sei alla ricerca di un' accelerazione nella tua carriera, ci sono un sacco di cose da imparare. "La nostra politica è letteralmente quella di assumere quanti più tecnici si trovino," dice Mark Zuckerberg, CEO di Facebook. "Il limite maggiore nel sistema è che non ci sono sufficienti persone che sono addestrate per queste capacità al giorno d'oggi."

Cosa è un Programma?

Scopri i mattoni del software e impara ciò che succede all'interno di un programma

Quale Python?

Python 2 e Python 3 sono entrambi usati correntemente. Python 3 è il futuro, e quindi ci rivolgiamo maggiormente verso di esso. Molti corsi insegnano ancora Python 2, e non è una cattiva scoprire da vicino le differenze tra i due: magpi.cc/2gP6zX3

Prima di procedere, diamo una occhiata a cosa effettivamente è un programma. Il vocabolario lo definisce come un insieme di istruzioni che consentono ad un computer di effettuare delle determinate operazioni.

Un programma per computer è come una ricetta. Contiene una lista di ingredienti, chiamata 'variabili', ed una lista di istruzioni chiamate 'dichiarazioni' o 'funzioni'. Segui le istruzioni della ricetta una riga alla volta ed ottieni una gustosa torta – e non si tratta di uno scherzo.

Il vero miracolo dei computer, tuttavia, è che possono ripetere le stesse cose in modo ripetitivo. Quindi si può costruire un dispositivo per preparare migliaia di torte senza neppure stancarsi. Un programma può contenere dei cicli che consentono di ripetere sempre le stesse istruzioni in continuazione.

I programmi possono inoltre prendere delle decisioni, ed eseguire istruzioni diverse a seconda del risultato. La tua ricetta può preparare l'impasto di una torta al cioccolato oppure di una deliziosa ciambella in funzione delle variabili (gli ingredienti) che contiene.

Compilato vs Interpretato

Python è un 'linguaggio interpretato'. Scrivi il codice e poi lanci il programma. Sotto sotto, viene tradotto al volo ed eseguito. Alcuni linguaggi di programmazione come C e Java, sono invece compilati. Tu scrivi il programma e poi lo compili ottenendo un file (scritto in codice macchina), e solo dopo lo puoi lanciare. E' un fuffa che puoi ignorare per ora.

Una cosa che può sorprendere quando si inizia a programmare è quanto poco è necessario conoscere per iniziare. Solo poche variabili ed un po' di funzioni, e puoi subito avere un computer che fa tutto il lavoro più impegnativo al tuo posto.

All'interno del tuo PI

Nel cuore del tuo Raspberry Pi ci sono miliardi di interruttori di tensione, chiamati anche cifre binarie (o 'bits' in forma abbreviata). Ce ne sono 8,589,934,592 in 1GB di RAM, per l' esattezza. Tutti questi interruttori possono essere impostati a valore alto oppure basso, che sono tipicamente rappresentati da 0 (per basso o off) ed 1 (per alto oppure on). Tutto quanto vedi sullo schermo, senti dagli altoparlanti e digiti sulla tastiera corrisponde a miliardi di interruttori che si posizionano su on e off.

Ovviamente, non è così semplice per gli uomini parlare direttamente con un computer . E' possibile utilizzare un linguaggio macchina ed inviare direttamente istruzioni binarie ad un computer, ma questo non è il punto di partenza per una persona sana di mente (o di fine, se vuole rimanere sana di mente).

Invece, utilizziamo un linguaggio di codici per programmare. Questo viene scritto utilizzando funzioni semplici da capire, come `print()`. Queste istruzioni vengono poi interpretate in linguaggio macchina, che può essere capito dal computer.

Noi useremo Python per imparare a programmare. È veramente un gran linguaggio di programmazione. Dispone di una ricca sintassi ma molto snella; non devi preoccuparti di parentesi graffe o altre diavolerie presenti nei linguaggi più complessi come java .

Con Python, crei il tuo programma, lo fai partire, ed è tutto fatto. Python è uno di quei linguaggi che maggiormente si trovano all'interno di The MagPi, e quindi impararlo qui ti aiuterà moltissimo nel comprendere i programmi utilizzati nei vari progetti.

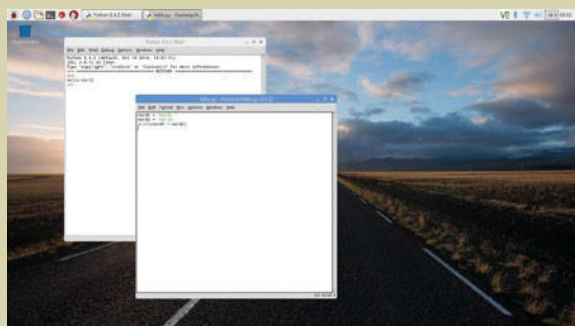
IDE e IDLE

Non è necessario scrivere i programmi Python usando un editor di testi e farli poi girare nel terminale. Invece, puoi utilizzare una soluzione tutta in uno, nota come 'IDE' (ambiente di sviluppo integrato).

Gli IDE abbinano un editor di testo con una funzione di avvio del programma. Spesso, comprendono utili caratteristiche come la ricerca di errori e il completamento automatico del testo.

Clicca **Menu > Programming >Python 3 (IDLE)**, ed otterrai una nuova finestra chiamata 'Python 3.4.2 Shell:'.
Questa Shell funziona proprio come la linea di comando Python. Digita **print("Ciao Mondo")** per vedere il messaggio.

Puoi anche creare programmi con un editor incorporato. Scegli **File > New File**. Digita questo programma nella finestra chiamata 'Untitled':

```
parola1 = "Ciao "
parola2 = "Mondo"
print(parola1 + parola2)
```

Sopra Python IDLE rende semplice creare un programma e farlo girare senza usare la linea di comando

Non dimenticare di includere lo spazio dopo 'Ciao'. Scegli **File > Save As** e salva come **ciao.py**. Ora premi **F5** sulla tastiera per far partire il programma. (O scegli **Run > Run Module**). Nella Shell verrà visualizzato 'Ciao Mondo'.

Il vantaggio di utilizzare Python IDLE sta nel fatto che tu puoi ispezionare il programma nella Shell. Digita **parola1**, e vedrai 'Ciao '. Digita **parola2** e vedrai 'Mondo'. Questa possibilità di ispezionare ed utilizzare le variabili nel tuo programma rende molto più semplice effettuare prove durante la programmazione e trovare errori (problemi nel tuo codice).

Perché Python?

Ci sono molti linguaggi di programmazione disponibili, e tutti offrono qualche cosa di speciale. Python
È una grande opzione per chi inizia. La sua sintassi (l'uso delle parole e dei simboli) è semplice da leggere. Può inoltre essere applicato in campo industriale, medico, scientifico, e risulta quindi essere ideale sia per chi inizia che per gli esperti.

Python nel terminale

Non hai bisogno di fare nulla per configurare Python sul tuo Raspberry Pi. Apri il terminale in Raspbian e digita **python --version**. Visualizzerà 'Python 2.7.9'. Digita **python3 --version** e vedrai 'Python 3.4.2'.

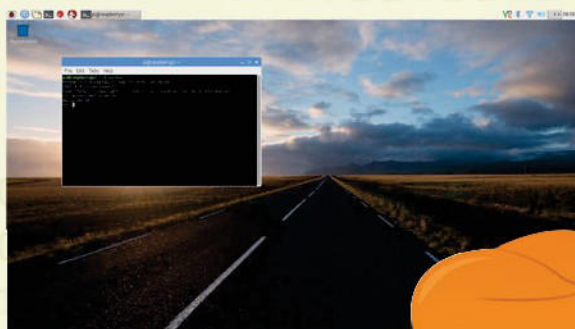
Utilizzeremo Python 3 in questa modalità (vedi il riquadro 'Quale Python?'). Puoi aprire Python 3 nel terminale digitando semplicemente **python3**.

Il prompt '$' della linea di comando sarà sostituito da '>>>'. Qui puoi inserire direttamente i comandi, proprio come fai nel terminale.

È tradizione battezzare ogni nuovo linguaggio visualizzando 'Hello World'. Digita **print("Ciao Mondo") e premei RETURN**. Vedrai comparire 'Ciao Mondo' nella linea seguente .

Usare la Shell viene chiamato Modalità Interattiva. Puoi interagire direttamente con il codice. E' comodo per fare calcoli matematici; inserire **1920 * 1080** e si otterrà in risposta: 2073600.

Il più delle volte creerai programmi in Python usando un normale editor di testo e salverai i file con l' estensione '.py'. Non usare un word processor come LibreOffice, però – inserirà della formattazione che si mischierà al codice, alterandolo.

Utilizza un editor di testo come Leafpad (**Menu > Accessories > Text Editor**). Qui puoi inserire il tuo codice, salvarlo come programma, e poi farlo girare nel terminale. Digitare **python3 nomeprogramma.py** sulla linea di comando per lanciarlo.

A sinistra Python è già preinstallato nel sistema operativo Raspbian e lo puoi utilizzare con la linea di comando

Variabili

Le variabili sono contenitori multiuso per memorizzare dati e oggetti

Tipi di variabile

Python ha cinque tipi di standard di dati:

- Numeri
- Stringhe
- Liste
- Tuple
- Dizionari

Foo bar?

Si incrociano spesso 'foo' e 'bar' quando si cerca di imparare il codice. Questi sono solo stupidi segnaposto e non significano nulla. Potrebbero essere zig e zag o bim e bam. Nessuno ne è abbastanza sicuro, ma potrebbero essere correlati all'espressione 'fubar' dalla guerra del Vietnam.

Se hai creato un progetto scientifico o un esperimento, puoi aver già incontrato delle variabili. Nella scienza, una variabile è un qualsiasi fattore che puoi controllare, cambiare o misurare.

Nella programmazione informatica, le variabili sono usate per memorizzare le cose nel tuo programma. Possono essere nomi, numeri, etichette, e tag: tutto quello di cui ha bisogno il tuo programma.

In Python, scrivi il nome della variabile, poi un solo segno di uguale e la parola, il numero o l'oggetto che ci vuoi mettere.

Digita questo codice direttamente nella Shell:

```
foo = 1
bar = 2
```

Ricorda: il nome della variabile è a sinistra, e quello che contiene è a destra. Immagina di avere due bicchieri di plastica, e tu hai scarabocchiato 'foo' sul primo e 'bar' sul secondo. Metti il numero 1 in foo e il numero 2 in bar.

Se vuoi conoscere nuovamente il numero, basta guardare nel bicchiere. In Python lo fai usando solo il nome della variabile

```
foo
bar
```

Tu puoi anche stampare delle variabili passandole alla funzione **print**

```
print(foo)
print(bar)
```

Le variabili possono anche essere utilizzati per contenere 'stringhe'. Queste sono gruppi di lettere (e altri caratteri) che formano parole, frasi o altri testi.

La creazione di una variabile stringa in Python è praticamente come per i numeri interi, tranne che il testo è racchiuso da apici (**' '**) o virgolette (**" "**).

Usare le virgolette rende più facile includere gli apostrofi, come ad esempio **print("Don't worry. Be Happy")**. Questa linea si romperebbe dopo 'Don' se si usassero gli apici – **print('Don't worry, be happy')** – quindi utilizza le virgolette, per ora.

Perché le variabili contano

Le variabili rendono molto più facile cambiare parti del tuo codice. Diciamo che hai un ottimo lavoro da programmatore alla a Nursery Rhymes Inc e hai scritto un classico (NdTrad: è una filastrocca inglese):

```
print("Polly accendi il bollitore")
print("Polly accendi il bollitore")
print("Polly accendi il bollitore")
print("Noi tutti abbiamo il te")
```

Arriva il capo del marketing entra e dice "i nostri dati affermano che Polly non è di tendenza con il millennio demografico." Tu dici "Huh!" e lui risponde "Cambia Polly in Dolly."

A questo punto è necessario scorrere tutte le linee di codice per modificare le variabili. Che scocciatura! Cosa sarebbe successo se avessi scritto migliaia di righe di codice e era necessario cambiarle tutte? Saresti rimasto lì tutta la settimana.

Con le variabili, la definisci una volta e poi la utilizzi nel codice. Così è già pronto per essere modificato

in ogni momento:

```
nome = "Polly"

print(nome + " accendi il bollitore")
print(nome + " accendi il bollitore")
print(nome + " accendi il bollitore")
print("Noi tutti abbiamo il te")
```

Questo codice stampa la stessa classica filastrocca. Ma se vuoi cambiare il nome del nostro protagonista, è sufficiente cambiarlo in un posto solo:

```
nome = "Dolly"
```

...e la poesia si aggiornerà su ogni linea.

Qual'è il tuo tipo?

Quando crei una variabile in Python, a questa viene assegnato automaticamente un tipo in base a quello che contiene. Puoi controllarlo utilizzando la funzione **type()**. Nell'interfaccia della shell, digita:

```
foo = "Dieci"
bar = 10
```

Ora usa la funzione **type()** per verificare il tipo di ciascuna variabile:

```
type(foo)
type(bar)
```

Il risultato sarà **<class 'str'>** per **foo**, e **<class 'int'>** per **bar**. Questo concetto è importante, perché differenti tipi lavorano insieme in modi diversi, e questi non sempre vanno bene insieme.

Per esempio, se sommi tra loro due stringhe, queste vengono combinate assieme:

```
nome = "Harry"
lavoro = "Mago"
print("Sei un " + lavoro + ", " + nome)
```

Questo stampa il messaggio "Sei un Mago, Harry". Le stringhe sono concatenate (che è un simpatico termine da programmatori per 'unite assieme'). I numeri, invece, funzionano in modo completamente diverso. Prova un po' di matematica:

```
numero1 = 6
numero2 = 9

print(numero1 + numero2)
```

Invece di concatenare 6 e 9 insieme e ottenere 69, Python esegue un po' di matematica, e ottieni '15' come risposta.

Type casting

Cosa succede quando si desidera aggiungere tra di loro una stringa e un numero intero?

```
nome = "Ben"
numero = 10
print(nome + numero)
```

Otterrai un messaggio di errore: 'TypeError:cannot concatenate 'str' and 'int' objects'. Capita questo errore perché Python non può sommare tra loro una stringa e un numero intero, perché lavorano diversamente. Ah, ma non correre! Puoi infatti moltiplicare stringhe e numeri:

```
print(nome * numero)
```

Verrà stampato 'Ben' dieci volte: otterrai quindi 'BenBenBenBenBenBenBenBenBenBen'.

Se desideri ottenere 'Ben10', avrai bisogno di convertire il numero intero in una stringa. Lo puoi fare utilizzando la funzione **str()** e mettendo il numero intero all'interno delle parentesi. Lo facciamo, e memorizziamo il risultato in una nuova variabile chiamata **numero_come_stringa**

```
numero_come_stringa = str(numero)
print(nome + numero_come_stringa)
```

Questo codice stamperà il nome 'Ben10'. Questo concetto è noto come 'type casting': convertire una variabile da un tipo a un altro.

Puoi anche convertire le stringhe in numeri interi usando la funzione **int()**. È particolarmente utile quando usi **input()**per ottenere un numero da parte dell'utente; l'input è memorizzato come una stringa. Creiamo un programma che chiede un numero e un esponente e elevi il numero alla potenza dell'esponente (utilizzando il simbolo '**'):

```
numero = input("Immetti un numero: ")
esponente = input("Immetti un esponente: ")
risultato = int(numero) ** int(esponente)
```

Le nostre prime due variabili, **numero** e **esponente**, sono stringhe, mentre la terza, **risultato**, è un numero intero. Possiamo stampare il risultato:

```
print(risultato)
```

Ma se noi vogliamo includere un messaggio, abbiamo bisogno che **risultato** sia di tipo stringa:

```
print(numero + " elevato alla potenza
   " + esponente + " è " + str(risultato))
```

Variabili, tipi, e type casting potrebbe essere difficili all'inizio. Python è molto più facile da usare perché cambia dinamicamente il tipo delle variabili per adattarle al contenuto. Tuttavia, ciò significa che devi fare un po' di attenzione.

Come chiamare una variabile ?

I nomi delle variabili devono essere in minuscolo e con parole separate da un trattino basso'_'. Possono includere numeri, ma devono iniziare con una lettera. Puoi chiamare le variabili come preferisci, ma c'è una piccola lista di parole chiave riservate che dovresti evitare (magpi.cc/2h7MH1y). È una buona idea nominarle in modo che risulterà ovvio quando le userai nel programma, come 'nome_studente' o 'eta_persona'.

Controllare il flusso

While & For

Fai fare al tuo programma tutto il lavoro sporco con i cicli While e For

Operatori di confronto

Questi operatori di confronto sono comunemente utilizzati in scelte condizionali per determinare se una determinata condizione è Vera o Falsa:

- `==` uguale
- `!=` non uguale
- `<` minore di
- `<=` minore o uguale
- `>` maggiore di
- `>=` maggiore o uguale
- `<>` minore o maggiore di

I computer sono fantastici perché a loro non dispiace fare la stessa cosa ripetutamente. La loro natura da gran lavoratori li rende perfetti per svolgere il lavoro sporco.

Quando prima stavamo dando un'occhiata alle variabili, abbiamo utilizzato la funzione print per visualizzare questa filastrocca:

```
print("Polly accendi il bollitore")
print("Polly accendi il bollitore")
print("Polly accendi il bollitore")
print("Noi tutti abbiamo il te")
```

Non ci è piaciuta la ripetizione di Polly e così l'abbiamo rimpiazzata con una variabile. Ma questo programma è folle anche per un altro motivo: devi scrivere lo stesso **`print`** per tre volte.

Utilizzeremo un ciclo per liberarci della ripetizione. Il primo ciclo di cui parleremo è il 'ciclo while'. In Python 3 IDLE, crea un nuovo file e salvalo come **polly.py**; immetti il programma che trovi dall'inizio della pagina successiva.

Iniziamo con due variabili:

```
nome = "Polly"
contatore = 0
```

Poi utilizziamo la dichiarazione **`while`** seguita dalla condizione: **`contatore < 3`**.

Nella riga successiva, premi la barra spaziatrice quattro volte per identare il codice. Non usare il tasto **TAB** (vedi il riquadro 'Tab o spazi?').

```
while contatore < 3:
    print(nome + " accendi il bollitore")
    contatore = contatore + 1
```

Il simbolo **<** significa 'minore di'. Esso controlla che quel che c'è a sinistra sia minore di quel che c'è a destra. In questo caso controlla che la variabile contatore (che parte da 0) sia minore di 3. Questa condizione è riconosciuta come 'Vera'; in caso contrario sarebbe 'Falsa'.

Infine digita l'ultima linea del codice:

```
print("Noi tutti abbiamo il te")
```

Salva e lancia il programma (premi F5). Stamperà a video 'Polly accendi il bollitore' 3 volte e poi 'Noi tutti abbiamo il te'.

While, condizione ed identazione

Ci sono tre cose qui: la dichiarazione while, la condizione e il testo identato, organizzate così:

```
while condizione:
    indentazione
```

Immagina una chiacchierata a tre tra gli oggetti del nostro programma **polly.py**:

Tab o spazi?

C'è un enorme dibattito tra i nerd quando si parla se usare tab o gli spazi nell'identare un programma. Le argomentazioni sono buone tra entrambe le parti e puoi impararle da questa clip della commedia ***Silicon Valley*** (magpi.cc/2gZde0M). Per il momento però usa gli spazi. Quando sarai un programmatore compulsivo potrai avere un parere sul tab.

While: "Hey Condizione! Qual'è il tuo stato?"
Condizione: "Vero! Il contatore è 0. E' minore di 3."
Indentazione: "OK, ragazzi. Stamperò a video 'Polly accendi il bollitore' e incrementerò il contatore di 1. Cosa viene dopo?"

While: "Hey Condizione. Qual'è il tuo stato?"
Condizione: "Vero! Il contatore è 1 ora."
Indent: "OK. Stampo un altro 'Polly accendi il bollitore' ed incremento il contatore di 1."

Questo continua fino a che il contatore non arriva a 3.

While: "Hey Condizione. Qual'è il tuo stato?"
Condition: "Falso! Il contatore è ora a 3, il quale non è minore di 3"
While: "OK ragazzi. Abbiamo finito!"

Il programma non esegue le istruzioni identate ma si sposta sul singolo **print** alla fine: 'Noi tutti abbiamo il te'.

For e le liste

Il prossimo tipo di ciclo è conosciuto come 'for'. È stato progettato per lavorare con le liste.

Le liste sono delle variabili che contengono più oggetti (stringhe, numeri, o anche altre variabili). Crea una lista mettendo gli oggetti tra parentesi quadre:

```
banana_splits = ["Bingo", "Fleegle",
                 "Drooper", "Snorky"]
```

Ora digita **banana_splits** nella Shell per visualizzare la lista.
Verranno mostrati i quattro nomi contenuti nelle parentesi quadre. Puoi accedere ad ogni oggetto individualmente usando il nome della variabile e parentesi quadre. Inserisci:

```
banana_splits[0]
```

...ed otterrai 'Bingo'. Gli elenchi Python hanno indice 0 il che significa che il primo oggetto della lista è [0]. Qui ci sono tutti gli oggetti. Digitali nella Shell per ottenere i loro nomi:

Polly.py

```
nome = "Polly"
contatore = 0

while contatore < 3:
    print(nome + " accendi il bollitore")
    contatore = contatore + 1

print("Tutti noi abbiamo il tè")
```

```
banana_splits[0] # "Bingo"
banana_splits[1] # "Fleegle"
banana_splits[2] # "Drooper"
banana_splits[3] # "Snorky"
```

Le liste con indice 0 possono confondere all'inizio. Devi solo ricordare che stai contando a partire da 0. Il ciclo for semplifica il modo di ripetere gli oggetti in una lista. Crea questo programma e salvalo come *<95o=<h9B*:

```
banana_splits = ["Bingo", "Fleegle",
                 "Drooper", "Snorky"]

for banana_split in banana_splits:
    print(banana_split)
```

Non importa che variabile usi in un ciclo for, basta che ti ricordi di usarla nell'identazione. Potresti usare:

```
for pippo in banana_splits:
    print(pippo)
```

E' comune utilizzare il plurale per nominare le liste (esempio 'nomi', 'pagine', e 'oggetti') ed usare la versione singolare nelle variabili: 'for nome in nomi', 'for pagina in pagine', e avanti così.

Cicli infiniti

Devi stare attento quando cambi il contatore in un ciclo while o rischi di ottenere un ciclo infinito. Se cancelli la linea **contatore = contatore + 1** dal nostro ciclo while, esso continuerà per sempre: non andrà mai sopra lo 0, di conseguenza il codice identato non si fermerà mai. Questo bug è conosciuto come 'ciclo infinito' ed è una cosa che è meglio non avere nei tuoi programmi.

Scelte Condizionali

Dona ai tuoi programmi un po' di cervello con le scelte condizionali

Operatori logici

Puoi combinare le condizioni tra loro utilizzando gli operatori logici.

and Entrambi gli operandi sono veri (a e b) sono Veri.
or Qualunque operando è vero (a o b) è Vero.
not Verifica lo stato di falso: not (a and b) è Vero se sia a che b sono Falsi.

I tuoi programmi stanno piano piano diventando più potenti. Abbiamo imparato come eseguire istruzioni in ordine procedurale, sostituire parti del programma con variabili ed eseguirle ciclicamente.

Ma un'altra parte importante della programmazione è quella chiamata 'scelte condizionali'. La scelta avviene quando il programma decide se deve fare qualcosa oppure no.

Ovviamente il programma non decide questo per capriccio: qui usiamo delle solide basi di logica.

L'inizio di tutto è la potente dichiarazione 'if'. Assomiglia ad un ciclo ma viene eseguita una volta sola. La dichiarazione 'if' chiede se una condizione è Vera. Se lo è allora esegue il codice identato:

```
if True:
    print("Ciao Mondo")
```

Esegui questo programma, visualizzerà 'Ciao Mondo'. Ora cambia la dichiarazione 'if' con False.

```
if False:
    print("Ciao Mondo")
```

...e non succederà nulla.

Ovviamente non puoi solo scrivere True o False. Devi invece creare una condizione che valuta il Vero ed il Falso; una comune è con il segno uguale (==). Essa controlla se quel che ha da ambo i suoi lati è identico. Crea un nuovo file ed inserisci il codice da **password1.py**. È un semplice programma che ti chiede di immettere una password, se digiti la password corretta, 'qwerty', ti verrà mostrato un messaggio di benvenuto, 'Benvenuto'.

Fai attenzione a non confondere l'operatore logico di equivalenza == con il singolo simbolo di uguale =. Mentre l'operatore di equivalenza controlla che i due lati siano uguali, il singolo uguale li fa diventare uguali. Confondere == e = è un errore comune tra i programmatori principianti.

What else

Dopo 'if', la prossima scelta condizionale da imparare è 'else'. Questo comando è complementare ad if e funziona da alternativa. Quando la condizione if è Vera la corrispettiva identazione viene eseguita; se invece è Falsa verranno eseguite le istruzioni di else.

```
if True:
   print("Eseguo la prima scelta")
else:
   print("Eseguo la seconda scelta")
```

Esegui questo programma e si verificherà la prima scelta. Ma se cambi da True a False:

```
if False:
   print("Eseguo la prima scelta")
else:
   print("Eseguo la seconda scelta")
```

...vedrai che verrà eseguita la seconda scelta. Usiamo questo per ampliare il nostro programma password. Digita il codice che trovi in **password2.py**.

Esegui di nuovo il programma. Se la password è corretta, ora otterrai un messaggio di benvenuto. Altrimenti verrà visualizzato 'password errata!'.

Elif

La terza scelta condizionale da conoscere è 'elif'. Questa dichiarazione sta per 'else if', e si trova a metà tra una dichiarazione if ed una else. Diamo ora un'occhiata ad una dichiarazione elif. Digita:

```
if False:
    print("Eseguo il 1 blocco di codice")
elif True:
    print("Eseguo il 2 blocco di codice")
else:
    print("Eseguo il 3 blocco di codice")
```

Esegui questo programma e vedrai che salterà la prima dichiarazione ma eseguirà l'elif. Vedrai comparire quindi "Eseguo il 2 blocco di codice".

La dichiarazione else non ha una condizione Vera o Falsa; viene eseguita solo se if ed elif non sono Vere. (Nota che qui, come sempre, la dichiarazione else è opzionale; puoi utilizzare solo if ed elif.)

Ma cosa succede se dichiari sia if che elif Vere? Prova e vedrai se verrà eseguito solo if , elif , oppure entrambe. Sperimenta rimuovendo la dichiarazione else e giocaci un po'. Ti aiuterà a capire le tre dichiarazioni if, elif ed else.

FizzBuzz

Ora ti mostreremo un programma comunemente usato nei colloqui per programmatori. E' un classico chiamato 'FizzBuzz', e ti mostrerà se hai davvero capito if, else ed elif.

Prima di tutto, devi conoscere l'operatore percentuale (%). Serve ad ottenere il resto di una divisione ed è simile all'operatore di divisione. Prendi questa funzione:

```
10 / 4 == 2.5
```

Se invece utilizziamo il percentuale, otteniamo:

```
10 % 4 == 2
```

Può essere utile in molti modi. Puoi usare `% 2` per capire se un numero è pari o dispari.

```
10 % 2 == 0 # questo è pari
11 % 2 == 1 # questo è dispari
```

Questo programma ritorna se un numero è pari o no:

```
numero = 10

if numero % 2 == 0:
    print("Il numero è dispari")
else:
    print("Il numero è pari")
```

OK – continuiamo con FizzBuzz.

Password.py

```
password = "qwerty"
attempt = input("Inserire password: ")

if attempt == password:
    print("Benvenuto")
```

Password2.py

```
password = "qwerty"
attempt = input("Inserire password: ")

if attempt == password:
    print("Benvenuto")
else:
    print("Password errata!")
```

Scrivere un FizzBuzz

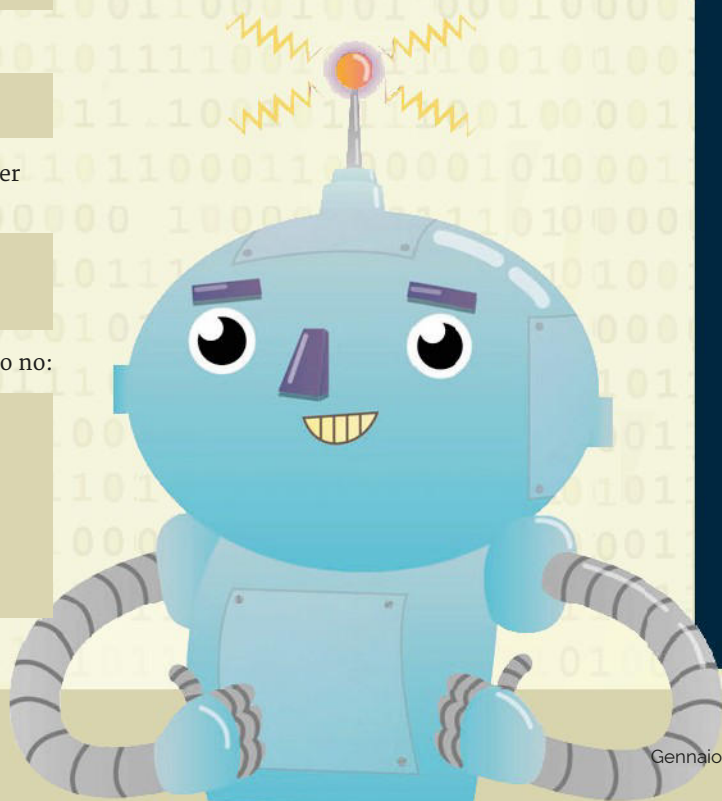
Il compito per il nostro FizzBuzz è stampare i numeri da 1 a 100. Se un numero è divisibile per 3 (come 3, 6, 9), verrà visualizzato 'Fizz' invece del numero, se è divisibile per 5, verrà visualizzato 'Buzz'

Ma se un numero è divisibile sia per 3 che per 5, come il numero 15, allora verrà visualizzato 'FizzBuzz'.

In FizzBuzz introduciamo anche un altro elemento: la dichiarazione 'and'. Essa controlla se due condizioni sono entrambe Vere: cioè che il numero possa essere diviso sia per 3 che per 5. Il risultato sarà vero solo se entrambe le condizioni sono vere.

Questi sono tre operatori logici principali: and, or e not. I primi due sono relativamente facili da capire ma l'operatore 'not' può confondere un po' all'inizio. Non preoccuparti troppo di questo, con un po' di pratica non sarà più un problema. Digita il codice **fizzbuzz.py** di pagina 25 per far pratica con l'utilizzo di if, else ed elif, e con gli operatori logici.

Commenti

Un marchio di un buon programmatore è l'utilizzo dei commenti nei propri programmi. I commenti vengono utilizzati per spiegare alcuni passaggi del programma agli umani. Vengono completamente ignorati dal computer.. In Python, si inizia un commento con il simbolo cancelletto (#). Può essere disposto su una singola riga oppure alla fine di una linea di codice.. Appena Python raggiunge il # , smette di tradurre quello che segue in codice macchina. I commenti aiutano gli altri utenti a leggere il tuo programma ma anche te a capire cosa stavi facendo (dopo che te ne sei dimenticato). E' quindi una buona abitudine usare i commenti nei tuoi programmi.

Creare le Funzioni

Creare i mattoni di codice e rendere più robusti i propri programmi

Ne hai fatta di strada dal tuo primo 'Ciao Mondo'. I tuoi programmi ora valutano condizioni e ciclano automaticamente.

Ora stai scrivendo programmi che sono conosciuti come 'Turing equivalenti', chiamati così da Alan Turing, il padre dell'ingegneria informatica e dell'intelligenza artificiale, che ha decodificato il linguaggio in codice tedesco Enigma nella Seconda Guerra Mondiale.

Ora approfondiremo un pochino le cose. Ti introdurremo nelle forme modulari chiamate funzioni.

Le funzioni sono blocchi di codice che scrivi una volta e le puoi ripetere ovunque. È un po' come poter scrivere un pezzo di testo una sola volta e poi incollarlo ogni volta che ti serve.

Individuare una funzione

Ci sono funzioni incorporate in Python che stai già utilizzando nei tuoi programmi. Comandi come **print()**, **len()**, e **type()** sono tutti funzioni. Sono facili da individuare: è un breve comando che inizia con una lettera minuscola ed è seguito da una coppia di parentesi '()'.

Documentazione di Python

Puoi navigare o scaricare una copia della documentazione di Python direttamente dal sito web python.org/doc. Python contiene già una vasta scelta di funzioni incorporate. Puoi trovare la lista completa sul sito della documentazione di Python (magpi.cc/2gPsGK3).

Usare le funzioni

Diamo un'occhiata alla funzione chiamata **abs()**. Significa 'assoluto' e da come risultato il valore assoluto di qualsiasi numero tu gli passi (ciò che gli passi è chiamato argomento).

Un numero assoluto è il positivo di ogni numero, se scrivi **abs(-2)** avrai come risultato 2. Prova questo nella Shell:

```
abs(2) # ritorna 2
abs(-2) # ritorna 2
```

Puoi memorizzare il risultato ottenuto come variabile:

```
numero_positivo = abs(-10)
```

Per noi è più semplice leggere una funzione al contrario, da destra a sinistra. Il valore viene inserito tra parentesi, poi la funzione lo manipola e ritorna il nuovo valore. Quest'ultimo viene passato a sinistra e memorizzato come variabile..

Definire una funzione

La cosa fantastica di Python è che non solo puoi usare le funzioni incorporate: ma puoi anche realizzare le tue.
Queste si chiamano 'funzioni definite dagli utenti'.

Le funzioni si creano utilizzando la parola chiave **def** seguita dal nome della funzione e dalle parentesi. All'interno delle parentesi elencherai i parametri. Sono la stessa cosa degli argomenti solo che, all'interno della definizione, vengono chiamati 'parametri'.

```
def function(parametro):
    return parametro
```

Questa nostra funzione, non fa nulla: semplicemente accetta un parametro e lo ritorna.

Alla fine della definizione della funzione ci sono i due punti (**:**). Il codice della funzione è identato da quattro spazi come in un ciclo o una scelta condizionale if/else.

Il codice all'interno dell'identazione viene eseguito quando richiami la funzione. Le funzioni tipicamente includono la dichiarazione **return** che restituisce una espressione.

Funzioni al lavoro

Ora creiamo una funzione che stampa a video il testo Buon Compleanno.

Digita il codice del listato **happy_birthday.py** e poi eseguilo. Nella Shell inserisci:

```
happy_birthday("Luca")
```

Questa chiamata di funzione passa la stringa 'Luca' come argomento. La stringa viene poi passata come parametro alla funzione e poi è disponibile per l'uso da parte del codice identato all'interno della funzione.

Dichiarazione Return

Molte funzioni non eseguono semplicemente un blocco di codice, ma restituiscono anche qualcosa alla chiamata della funzione.

Lo abbiamo visto nella funzione **abs()**, che ritorna il valore assoluto di un numero. Che può essere memorizzato in una variabile.

In effetti, ricreeremo la funzione **abs()** così puoi vedere come funziona dietro alle quinte.

In matematica puoi invertire il valore positivo o negativo moltiplicandolo per il numero negativo -1 , come qui:

```
10 * -1 = -10
-10 * -1 = 10
```

Dobbiamo creare una funzione che prende un numero come parametro e controlla se esso è negativo. Se lo è, viene moltiplicato per -1; se invece è positivo, viene restituito così com'è. Chiameremo la nostra funzione **absolute()**.

Digita il codice contenuto in **absolute.py**. Quando il programma raggiunge la dichiarazione **return** viene restituito il valore del numero (o da solo o moltiplicato per -1). Viene terminata poi la funzione.

Esegui il programma **absolute.py** e digita questo nella Shell:

```
absolute(10)
absolute(-10)
```

Il nostro ultimo listato di programma è un classico conosciuto come 'FizzBuzz'; come menzionato a pagina 23, ti aiuterà a capire if, else, and elif.

Ti serve anche conoscere l'operatore percentuale (**%**) per FizzBuzz. Questo operatore ritorna il resto di una divisione. Se non sai come un modulo opera, guarda questo video (**magpi.cc/2h5XNRO**).

Ora smanetta con il programma **fizzbuzz.py**.

Happy_birthday.py

```
def happy_birthday(nome):
    contatore = 0
    while contatore < 4:
        if contatore != 2:
            print("Buon compleanno a te")
        else:
            print("Buon compleanno caro " + nome)
        contatore += 1
```

Absolute.py

```
def absolute(numero):
    if numero < 0:
        return numero * -1
    else:
        return numero
```

Fizzbuzz.py

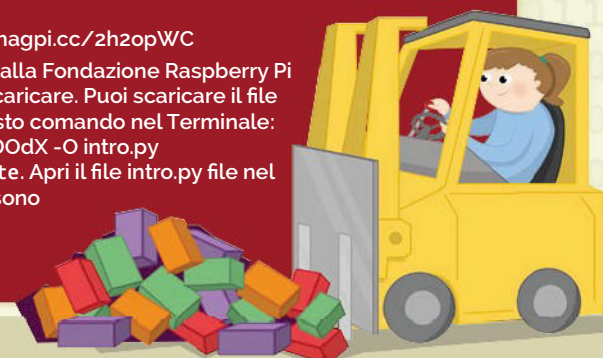
```
contatore = 0
fine = 100

while contatore < fine:
    if contatore % 5 == 0 and contatore % 3 == 0:
        print("FizzBuzz")
    elif contatore % 3 == 0:
        print("Fizz")
    elif contatore % 5 == 0:
        print("Buzz")
    else:
        print(contatore)

    contatore += 1
```

Per proseguire

Ecco un po' di risorse che potrebbero esserti utili.

GPIO Zero Essentials – magpi.cc/2bA3ZP7
Questo libro della serie "Essentials guide" spiega come la libreria Python GPIO Zero fornisca l' accesso a moltissime funzionalità. Queste vengono utilizzate per connettere componenti elettronici alla tua Raspberry Pi tramite i pin del GPIO.

FutureLearn – magpi.cc/2h5Sthf
La Fondazione Raspberry Pi ha due nuovi corsi online: Teaching Physical Computing with Raspberry Pi and Python, e Teaching Programming in Primary Schools.

Imparare Python – magpi.cc/2h2opWC
Questo tutorial fornito dalla Fondazione Raspberry Pi contiene file che puoi scaricare. Puoi scaricare il file intro.py utilizzando questo comando nel Terminale: wget http://goo.gl/oZDOdX -O intro.py `--no-check-certificate`. Apri il file intro.py file nel IDLE; tutte le istruzioni sono all'interno del file.

Importare il Codice

Sali sulle spalle dei giganti importando il codice degli altri programmatori

Pygame

Se vuoi saperne di più su Pygame, dai un'occhiata a ***Make Games With Python***, la nostra Essentials Guide al modulo Pygame.. magpi.cc/2h2m0vh

Questo è il mondo moderno, non ti viene richiesto di fare tutto il lavoro da solo. Ti affiderai, invece, spesso agli altri programmatori che hanno già fatto tutto il lavoro di base per te.

I tuoi programmi possono importare i codici creati da altre persone utilizzando la dichiarazione **import,** questa ti permette di importare i loro moduli ed utilizzare le loro funzioni – solo che ora si chiameranno 'metodi'.

Importa il modulo da linea di comando e poi si accedi alle funzioni con la notazione puntata. Questa è dove indichi il modulo seguito da un punto (.), e poi dal metodo.

Un modulo di uso comune è **math**. Esso ti permette di accedere a molti metodi matematici. Apri una Shell Python e digita:

```
import math
```

Ora hai accesso a tutti i metodi in **math**.
Non noterai nessuna differenza ma se digiti:

```
type(math)
```

...ti dirà '<class 'module'>'. Prova la notazione puntata ora. Digita **math** seguito da un punto e il nome di un metodo (funzione) che vuoi usare:

```
math.sqrt(16)
```

Fornisce la radice quadrata di 16, che è 4.

Alcuni metodi hanno più di un argomento. Il metodo **math.pow()** eleva un numero a una potenza

```
math.pow(64,3)
```

Il risultato è 262144.0.

Puoi anche accedere ai valori di costanti da un modulo, le quali sono delle variabili contenute nel modulo con un valore fisso.
Sono come le funzioni/metodi, ma senza parentesi.

```
math.pi
```

Fornisce la costante pi greco con 15 decimali: 3.141592653589793.

```
math.e
```

Questo invece è il numero di Eulero con 15 decimali: 2.718281828459045.

E' anche possibile importare metodi e costanti usando **from**. Questo ti permette di utilizzarli senza la notazione puntata (come le funzioni regolari). Per esempio:

```
from math import pi
from math import e
from math import pow
```

Ora, ogni volta che digiti *900* *, otterrai pi greco o il numero di Eulero. Puoi anche usare **pow()** come una normale funzione. Puoi anche cambiare il nome della funzione al momento dell'importazione con **as**:

```
from math import pi as p
```

Ora quando digiterai **p** otterrai pi greco con 15 decimali. Non impazzire rinominando le funzioni con **as**, ma è comune vedere alcuni metodi e costanti importati come lettere singole.

Creando le tue funzioni e importando quelle create dagli altri puoi migliorare di molto le capacità dei tuoi programmi.

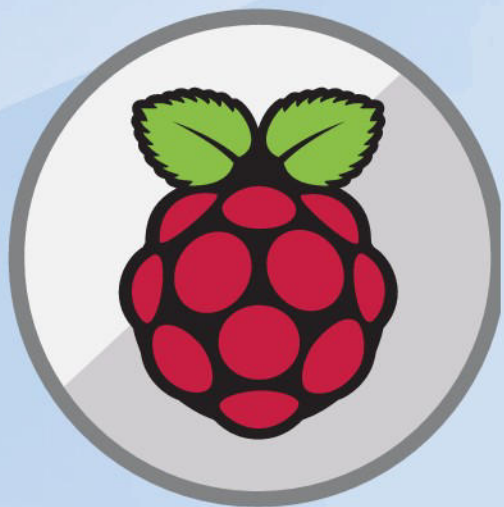
Ora utilizzeremo tutto quello che abbiamo imparato per creare il gioco Pong, che è uno dei primi videogiochi al mondo.

Scrivi attentamente il codice listato in **pong.py**. Ci troverai variabili, funzioni, cicli e scelte condizionali: tutto quello di cui abbiamo parlato prima. In teoria ora sarai in grado di decifrare la maggior parte di questo codice.

Se vuoi approfondire di più su Pong, questo programma è simile ad una versione Pygame di Trever Appleton (**magpi.cc/2hgkOUX**). La sua versione ha una tabella dei punteggi e un codice più avanzato. Noi abbiamo semplificato il nostro in modo che sia più facile iniziare a imparare con esso.

Speriamo che questa non sia la fine della tua esperienza con Python, o con la programmazione. Ci sono molte posti dove puoi imparare l'arte di programmare. E noi avremo altre risorse sulla programmazione in ogni numero di *The MagPi*.

Pong.py

```
01. import pygame, sys
02. from pygame.locals import *
03.
04. # Imposta variabili di gioco
05. window_width = 400
06. window_height = 300
07. line_thickness = 10
08. paddle_size = 50 # riduci questo per un gioco piu difficile
09. paddle_offset = 20
10.
11. # Imposta variabili colore
12. black = (0  ,0  ,0  ) # Le variabili tra parentesi sono 'tuple'
13. white = (255,255,255) # le tuple sono come le liste, ma i valori
    sono immutabili
14.
15. # Variabili palla (Coordinate cartesiane x,y)
16. # Posizione di partenza a metà della larghezza e altezza
17. ballX = window_width/2 - line_thickness/2
18. ballY = window_height/2 - line_thickness/2
19.
20. # Variabli per tracciare la direzione della palla
21. ballDirX = -1 ## -1 = left 1 = right
22. ballDirY = -1 ## -1 = up 1 = down
23.
24. # Posizione di partenza al centro dell' arena di gioco
25. playerOnePosition = (window_height - paddle_size) /2
26. playerTwoPosition = (window_height - paddle_size) /2
27.
28. # Crea rettangoli per la palla e le palette
29. paddle1 = pygame.Rect(paddle_offset,playerOnePosition, line_
    thickness,paddle_size)
30. paddle2 = pygame.Rect(window_width - paddle_offset - line_
    thickness, playerTwoPosition, line_thickness,paddle_size)
31. ball = pygame.Rect(ballX, ballY, line_thickness, line_thickness)
32.
33. # Funzione per disegnare l'arena di gioco
34. def drawArena():
35.     screen.fill((0,0,0))
36.     # Disegna il contorno dell' arena
37.     pygame.draw.rect(screen, white, (
    (0,0),(window_width,window_height)), line_thickness*2)
38.     # Disegna linea centrale
39.     pygame.draw.line(screen, white, (
    (int(window_width/2)),0),((int(window_width/2)),window_height), (
    int(line_thickness/4)))
40.
41. # Funzione per disegnare le palette
42. def drawPaddle(paddle):
43.     # Ferma la paletta se si sposta troppo in basso
44.     if paddle.bottom > window_height - line_thickness:
45.         paddle.bottom = window_height- line_thickness
46.     # Ferma la paletta se si sposta troppo in alto
47.     elif paddle.top < line_thickness:
48.         paddle.top = line_thickness
49.     # Disegna paletta
50.     pygame.draw.rect(screen, white, paddle)
51.
52. # Funzione per disegnare la palla
53. def drawBall(ball):
54.     pygame.draw.rect(screen, white, ball)
55.
56. # Funzione per muovere la palla
57. def moveBall(ball, ballDirX, ballDirY):
58.     ball.x += ballDirX
59.     ball.y += ballDirY
60.     return ball # returns new position
61.
62. # Funzione che controlla le collisioni col muro e cambia la
    direzione della palla
63. def checkEdgeCollision(ball, ballDirX, ballDirY):
64.     if ball.top == (line_thickness) or ball.bottom == (window_
height - line_thickness):
65.         ballDirY = ballDirY * -1
66.     if ball.left == (line_thickness) or ball.
right == (window_width - line_thickness):
67.         ballDirX = ballDirX * -1
68.     return ballDirX, ballDirY # ritorna nuova direzione
69.
70. # Funzione che controlla se la palla colpisce la paletta
71. def checkHitBall(ball, paddle1, paddle2, ballDirX):
72.     if ballDirX == -1 and paddle1.right == ball.left and
paddle1.top < ball.top and paddle1.bottom > ball.bottom:
73.         return -1 # ritorna nuova direzione (destra)
74.     elif ballDirX == 1 and paddle2.left == ball.right and
paddle2.top < ball.top and paddle2.bottom > ball.bottom:
75.         return -1 # ritorna nuova direzione (destra)
76.     else:
77.         return 1 # ritorna nuova direzione (sinistra)
78.
79. # Funzione per intelligenza artificiale giocatore del computer
80. def artificialIntelligence(ball, ballDirX, paddle2):
81.     # la palla si allontana dalla paletta, muovila al centro
82.     if ballDirX == -1:
83.         if paddle2.centery < (window_height/2):
84.             paddle2.y += 1
85.         elif paddle2.centery > (window_height/2):
86.             paddle2.y -= 1
87.     # Palla in movimento vero la paletta, traccia il movimento
88.     elif ballDirX == 1:
89.         if paddle2.centery < ball.centery:
90.             paddle2.y += 1
91.         else:
92.             paddle2.y -=1
93.     return paddle2
94.
95. # Inizializza la finestra
96. screen = pygame.display.set_mode((window_width,window_height))
97. pygame.display.set_caption('Pong') # Mostra nella finestra
98.
99. # Disegna l'arena e le palette
100. drawArena()
101. drawPaddle(paddle1)
102. drawPaddle(paddle2)
103. drawBall(ball)
104.
105. # Rendi invisibile il cursore
106. pygame.mouse.set_visible(0)
107.
108. # Il gioco principale gira in questo ciclo
109. while True: # Ciclo infinito. Premi Ctrl-C per uscire dal gioco
110.     for event in pygame.event.get():
111.         if event.type == QUIT:
112.             pygame.quit()
113.             sys.exit()
114.         # Movimenti del mouse
115.         elif event.type == MOUSEMOTION:
116.             mousex, mousey = event.pos
117.             paddle1.y = mousey
118.
119.     drawArena()
120.     drawPaddle(paddle1)
121.     drawPaddle(paddle2)
122.     drawBall(ball)
123.
124.     ball = moveBall(ball, ballDirX, ballDirY)
125.     ballDirX, ballDirY = checkEdgeCollision(
ball, ballDirX, ballDirY)
126.     ballDirX = ballDirX * checkHitBall(
ball, paddle1, paddle2, ballDirX)
127.     paddle2 = artificialIntelligence (ball, ballDirX, paddle2)
128.     pygame.display.update()
129.
```

LANCIA PIXEL SUL TUO COMPUTER

Il desktop **PIXEL** è ora disponibile per computer Intel: usa il DVD allegato gratuitamente questo mese per creare una chiavetta USB e impostare sul tuo portatile la migliore interfaccia desktop

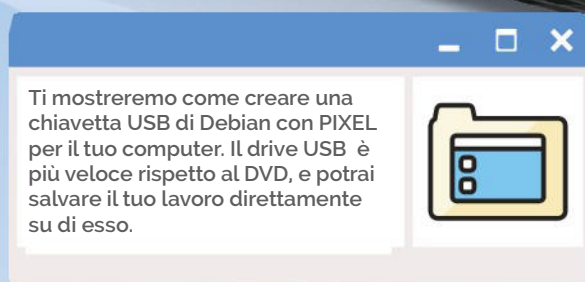
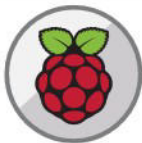
Il Raspberry Pi è di gran lunga il nostro computer preferito. Quindi, siamo stati lieti di scoprire che la Fondazione Raspberry Pi stava realizzando una versione di Debian con PIXEL per macchine Intel.

Con la versione cartacea di questo mese di *The MagPi*, troverai un DVD gratuito. Caricalo sul tuo computer portatile o desktop e lo renderai tale e quale a un Raspberry Pi, nell'utilizzo.

Se non hai il DVD di Debian con PIXEL, non preoccuparti. La Fondazione Raspberry Pi ha reso Debian con PIXEL disponibile al download come immagine file ISO.

Potrai utilizzare il file ISO per creare il tuo DVD di Debian con PIXEL, o per creare da solo una chiavetta USB avviabile che ti permetta di eseguire il desktop PIXEL sul tuo computer.

In questo speciale, ti mostreremo come avviare il computer in Debian con PIXEL e cosa potrai fare con il sistema operativo sul computer di casa. Daremo anche uno sguardo al salvataggio dei file e alla creazione di una chiavetta USB che potrai utilizzare per fare il boot in PIXEL su qualsiasi computer.

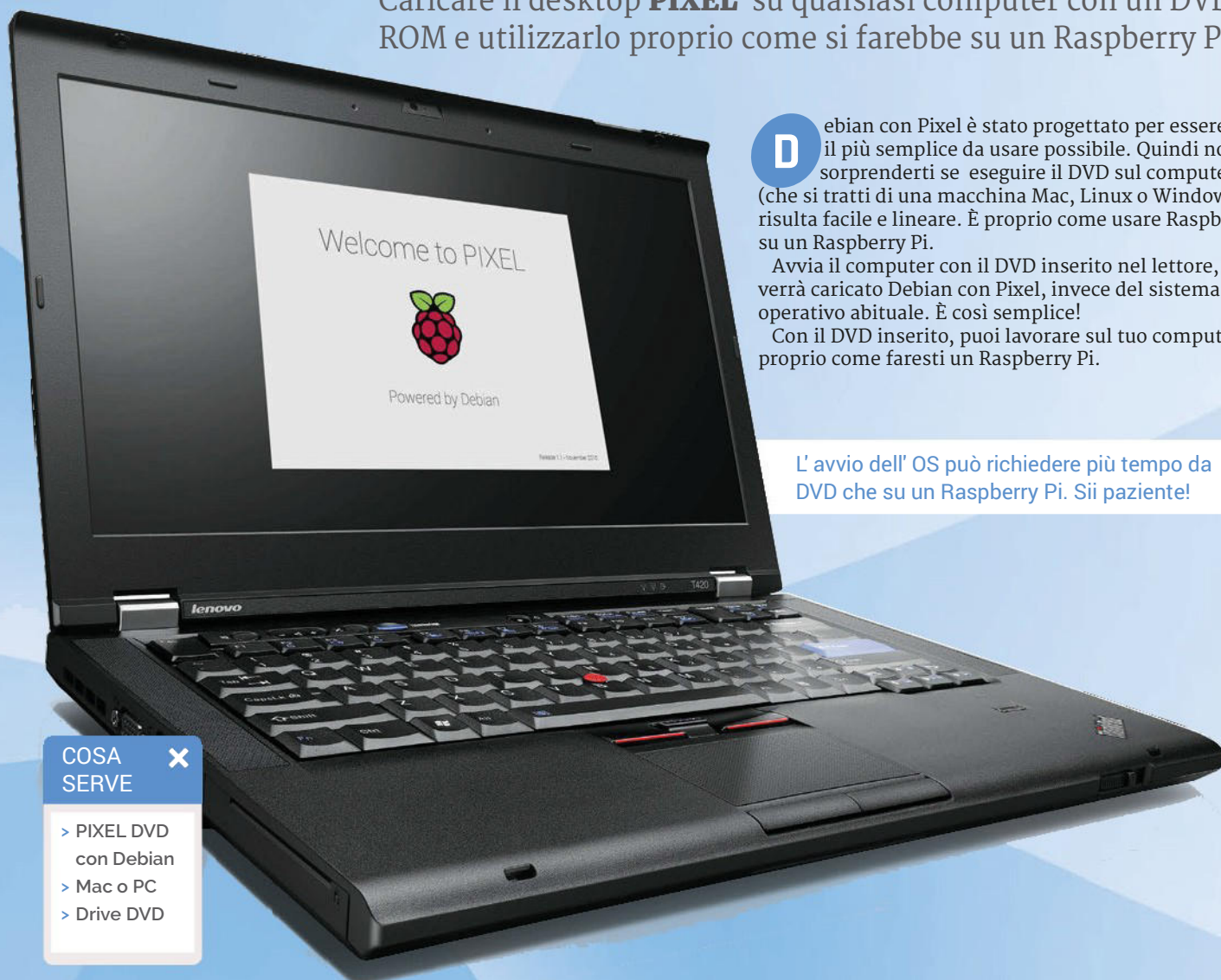
Ti mostreremo come creare una chiavetta USB di Debian con PIXEL per il tuo computer. Il drive USB è più veloce rispetto al DVD, e potrai salvare il tuo lavoro direttamente su di esso.

IL DESKTOP PIXEL

Il desktop PIXEL gira sul tuo computer, esattamente come su Raspberry Pi. Si trova sopra Debian, lo stesso sistema operativo Linux che viene compilato per ARM per creare Raspbian

DRIVE DVD

Carica Debian con PIXEL usando il DVD-ROM. Funziona direttamente da DVD e non interferirà con il tuo sistema operativo principale. Rimuovi il DVD e riavvia il computer quando hai finito. Il computer si riavvierà con il tuo consueto sistema operativo.

AVVIA IL COMPUTER CON IL DVD GRATUITO

Caricare il desktop **PIXEL** su qualsiasi computer con un DVD-ROM e utilizzarlo proprio come si farebbe su un Raspberry Pi

Debian con Pixel è stato progettato per essere il più semplice da usare possibile. Quindi non sorprenderti se eseguire il DVD sul computer (che si tratti di una macchina Mac, Linux o Windows) risulta facile e lineare. È proprio come usare Raspbian su un Raspberry Pi.

Avvia il computer con il DVD inserito nel lettore, e verrà caricato Debian con Pixel, invece del sistema operativo abituale. È così semplice!

Con il DVD inserito, puoi lavorare sul tuo computer, proprio come faresti un Raspberry Pi.

L' avvio dell' OS può richiedere più tempo da DVD che su un Raspberry Pi. Sii paziente!

COSA SERVE

- PIXEL DVD con Debian
- Mac o PC
- Drive DVD

Avvio

Accendi il computer e inserisci il DVD nel lettore. Non c'è bisogno di dire che devi utilizzare un computer dotato di lettore DVD. Se il tuo computer ne è sprovvisto, allora dovrai creare una chiavetta USB da cui fare il Boot con Debian con PIXEL – vedi pagina 76.

Se stai utilizzando hardware tipo PC, non importa se normalmente avvii direttamente Windows o Linux. La maggior parte dei Personal Computer è configurata per l'avvio da DVD prima del disco fisso. Dovresti sentire il disco girare nel lettore e vedere il messaggio di benvenuto 'Welcome to PIXEL powered by Debian' invece del solito sistema operativo.

Se stai utilizzando un computer Apple Mac, puoi tenere premuto il tasto **G** quando senti il suono 'bong' di avvio di Apple, oppure il tasto ALT e scegliere DVD come disco di avvio.

Il processo di boot

Dopo poco tempo vedrai una videata che mostra 'Debian (Jessie) with PIXEL i386'. Vedrai anche la scritta 'Press Esc for options' ed un timer con il conto alla rovescia.

BOOT MENU

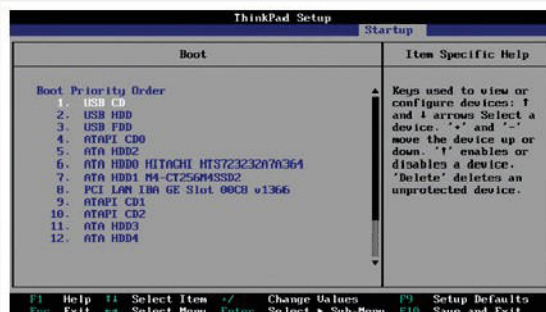
Premendo ESC durante l'avvio di Debian con PIXEL, viene mostrato il menu di avvio. In questo caso dovresti vedere due opzioni: 'Run Debian' e 'Run Debian without Persistence'. Se non premi ESC la prima è l'opzione predefinita. In questo speciale parleremo molto di 'Persistence'. Questa funzione serve ad abilitare il salvataggio delle modifiche al sistema operativo attuate quando si lavora da DVD o da chiavetta USB avviabile.

Selezionare 'Debian without Persistence' potrebbe semplificare l'avvio di Debian, non sarai però abilitato a salvare le modifiche apportate ai file o al sistema operativo.

Premi TAB per visualizzare la linea di comando di boot, in questo modo potrai aggiungere o togliere opzioni specifiche. Aggiungendo `nomodeset` dopo 686-pae può aiutare alcuni vecchi Mac ad avviarsi.

Una volta che sei soddisfatto delle opzioni di avvio, premi ENTER per avviare il DVD.

BIOS PC

Abbiamo verificato che Debian con PIXEL è affidabile sul nostro PC di test. Se non si dovesse avviare, probabilmente devi modificare le impostazioni del BIOS (Basic Input/Output System). Premi F1 sulla tastiera (o segui le istruzioni sullo schermo) per accedere alle impostazioni del BIOS dopo l'accensione. Si tratta delle impostazioni hardware predefinite del tuo computer (impostazioni che non dipendono dal sistema operativo). Usa i tasti direzione per spostarti e cerca Startup e Boot. Sul nostro ThinkPad l'opzione si chiama Boot Priority Order. Usa i tasti + e – per spostare il DVD nella prima posizione. Seleziona Save and Exit, quindi riavvia il sistema per riprovare

Non premere nulla. Quando il timer arriva a zero vedrai il messaggio 'Loading, please wait'.

Devi essere paziente quando carichi da DVD, potrebbe metterci parecchio prima di avviarsi, a seconda della velocità del lettore. Sii paziente.

Logging in

Durante il processo di avvio vedrai apparire la schermata di login di Raspberry. Non inserire nulla; verrai autenticato automaticamente e portato alla interfaccia desktop PIXEL. L'utente predefinito è **pi**, e la password è **raspberry**.

> "Puoi lavorare sul tuo computer proprio come faresti su un Raspberry Pi."

Usare Debian con PIXEL

Adesso puoi utilizzare Debian con PIXEL sul tuo computer proprio come faresti su un Raspberry Pi. Attenzione che tutti i file che crei non verranno salvati quando spegni. Scegli **Menu > Shutdown > Shutdown** per spegnere il sistema operativo quando hai finito.

USARE IL DVD

Utilizza **Debian con PIXEL** sul tuo computer

FORNITO CON PROGRAMMI

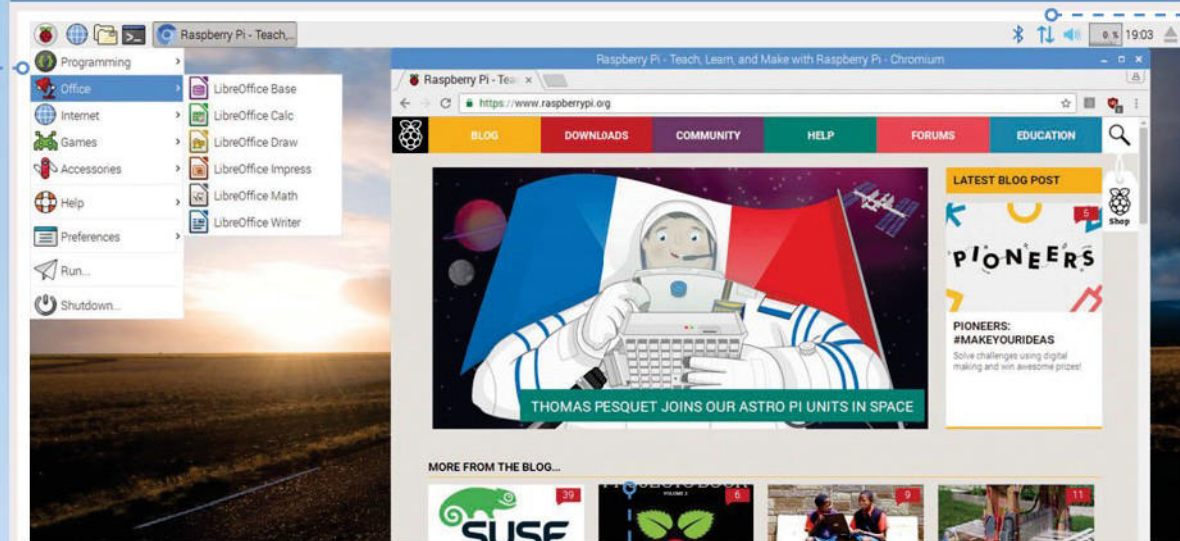
Il team ha selezionato il miglior software disponibile per l'ufficio e la programmazione. Questi software rendono Debian con PIXEL un sistema ideale e creativo per imparare.

NETWORK

Puoi connettere l'OS a una rete wireless locale e usare il browser web di default Chromium (con Flash) per accedere alle risorse online.

L'INTERFACCIA DESKTOP PIXEL

IL DESKTOP DI PIXEL

Il desktop di PIXEL su di un comune computer appare e funziona quasi allo stesso modo che sul Raspberry Pi. Questo lo rende ideale per imparare a programmare.

Quando Debian con PIXEL si è caricato potrai usare il tuo computer come se stessi usando un Raspberry Pi reale.

Inizia cliccando l'icona del Menu in altro a sinistra per accedere al software preinstallato. Qui troverai la migliore collezione di strumenti per lo sviluppo e per l'apprendimento. Debian con PIXEL comprende anche LibreOffice, una valida alternativa a Microsoft Office.

Ma sono gli strumenti per lo sviluppo il suo punto forte. Hai a disposizione programmi come Greenfoot Java, Scratch, e Sonic Pi. C'è persino un emulatore del Sense HAT ed i link a tutte le risorse e la documentazione online di Raspberry Pi

Vai online

La prima cosa che vorrai fare è connettere Debian con PIXEL ad internet. Clicca sull'icona Network nella barra del menu per visualizzare le reti wireless locali. Scegli una rete ed inserisci la password. (la 'Pre-Shared Key'). In alternativa, se il tuo computer ha una presa Ethernet, puoi usare un cavo Ethernet per connetterti direttamente alla rete o al router di casa.

Quando sei online, puoi navigare in internet utilizzando il browser Chrome con supporto Flash già preinstallato. Puoi anche scaricare software Debian utilizzando APT.

I programmi che installi usando APT vengono salvati nella memoria e possono essere utilizzati fino a che non spegni il computer.

Quando spegni Debian, tutti i file che hai creato (e i programmi che hai installato) vengono cancellati dal sistema. Questa pulizia può anche essere utile.

Senza un disco persistente, Debian con PIXEL parte come se fosse stato appena installato ogni volta, rendendolo un attrezzo utile per l'insegnamento. Però fai attenzione a non perdere qualche file importante.

Disco persistente

Se vuoi salvare i file e mantenere i programmi che hai installato, avrai bisogno di un disco persistente.

Questa unità lavora in tandem con il tuo DVD. Il DVD è usato per avviare il computer, e i file che salvi nella cartella home vengono memorizzati nella penna USB, ma il tutto lavora come fosse una singola unità.

Creare un disco persistente è piuttosto semplice. Basta formattare una penna USB in formato Linux (di solito ext4) e etichettarlo come **persistence**.

Dopo aver formattato l'unità devi creare un file di configurazione nella radice del disco. Il file è chiamato **persistence.conf** e contiene il testo: **/ union**. Questo file di configurazione comunica alla partizione USB di lavorare in unione con il DVD.

Formatta il disco

Il modo più semplice per formattare un hard drive è con l'app GParted. Apri il terminale e digita **sudo apt install gparted**. Quando ha finito, digita: **sudo gparted** per eseguire il programma con i permessi da super user.

Clicca su Disks menu in alto a destra in Gparted per vedere i tuoi hard drive (dovrebbe chiamarsi /dev/sda). Dovrebbe esserci un'opzione per ogni unità del tuo computer.

Ora connetti la penna USB e seleziona **GParted > Refresh Devices** (o premi **CTRL+R**). Clicca Disks menu di nuovo, e vedrai comparire la nuova scelta.

Tipicamente sarà il dispositivo con meno spazio disponibile (in Gb). Fai attenzione a selezionare l'unità USB e non il tuo disco fisso principale.

DISCO PERSISTENTE

Installa GParted per formattare una penna USB e creare un disco persistente. Questo disco viene usato per salvare i file e le modifiche che apporti al sistema operativo Debian.

LA STORIA DIETRO DEBIAN CON PIXEL

Ci siamo incontrati con Eben Upton, co-fondatore di Raspberry Pi e CEO di Raspberry Pi Trading, per parlare della creazione di Debian con PIXEL per macchine Intel.

"Immagino che la vera domanda sia 'perché non lo abbiamo fatto prima?'," inizia Eben. "Siamo abituati a sentire persone che ci dicono cose come: 'perché non fate software?' e 'tutti hanno un PC'."

La nostra risposta di allora era: 'non tutti hanno un PC' e 'le persone hanno paura di danneggiarli e per questo non vogliono sporcarsi le mani'.

Così la Raspberry Pi Foundation ha per molto tempo concentrato i suoi sforzi nel realizzare hardware e il software usato su Raspberry Pi è stato generico per la maggior parte di quel tempo. "Poi, dopo un po', Simon [Long] si è unito a noi," prosegue Eben.

Simon ha lavorato moltissimo per trasformare il desktop di serie di Raspbian nell'esperienza più accattivante che offre PIXEL.

La cosa superba d è la "curation" spiega Eben, cioè come il software sia stato accuratamente selezionato, ottimizzato, catalogato, con misura equilibrio e armonia.

Software come Flash, Chromium, Java, e LibreOffice richiedono molto lavoro per funzionare bene nel contesto. "Sta sempre più diventando l'ambiente perfetto sia per la produttività che per imparare a programmare"

"Abbiamo ora un software piuttosto interessante. La questione ora è: 'perché costringiamo le persone ad acquistare un Raspberry Pi per usarlo?'

"E così abbiamo chiuso il cerchio," Conclude Eben. "Vogliamo provare a vedere se alle persone piace."

Come prima cosa, clicca con il tasto destro la lunga barra orizzontale e scegli l'opzione Rimozione sicura (Unmount – se è disponibile). Poi, sempre con il tasto destro clicca su ogni partizione presente nella lista e seleziona Elimina (Delete) fino a che non vedi una barra orizzontale grigia con 'non allocato' (unallocate).

Scegli **Partition > New**. Inserisci persistence nel campo label e clicca OK.

Infine, clicca sull'icona visto verde (Apply All Operations – applica tutte le operazioni) e clicca su Applica (Apply) nella finestra che compare.

Configura il disco

Ora dobbiamo aggiungere il file **persistence.conf** nella radice del nostro disco formattato. Rimuovi e reinserisci la chiavetta USB.

Digita `cd /media/pi/` e `sudo chown pi persistence` per cambiare il proprietario del disco.

Poi, digita `cd persistence/` per spostarti nella radice del disco USB, ed infine `sudo echo / union > persistence.conf`.

Il comando echo scrive '/ union' in un file chiamato **persistence.conf** nella radice del tuo disco. Puoi verificarlo usando `cat persistence.conf`.

Ora tutto quello che devi fare è riavviare il computer. Digita `sudo reboot`. Il computer si avvierà dal DVD, ed utilizzerà la penna USB come disco persistente.

REALIZZA UNA CHIAVE USB CON DEBIAN CON PIXEL

Che cosa succede se non hai un lettore DVD? Non preoccuparti: è possibile creare una chiave USB avviabile...

COSA SERVE

- Chiave USB
- File ISO di DEBIAN con PIXEL

Molti computer recenti non sono dotati di lettore DVD. Quindi - cosa succede se desideri eseguire Debian con PIXEL sul tuo computer, ma non disponi di un lettore?

La risposta è piuttosto semplice: creare una chiave USB avviabile dal file ISO di Debian con PIXEL.

La maggior parte dei computer moderni possono avviarsi direttamente da un drive USB. Questa unità è più veloce e un po' più affidabile rispetto all'utilizzo di un DVD (e i computer portatili più vecchi possono avere dei lettori unità DVD a pezzi).

È possibile partizionare un disco USB per contenere sia Debian con pixel e che il disco di persistenza. È come avere un Raspberry Pi in tasca che si può avviare praticamente su qualsiasi altro computer.

Suona divertente? Facciamo uno!

BARRA DI MENU MANCANTE

Abbiamo riscontrato che, a volte, la barra dei menu di PIXEL, non appare. In questo caso, premi CTRL + ALT + F1 per passare a un'altra sessione (inserisci pi e raspberry per effettuare il login). Ora inserisci `sudo pkill lxsession` per riavviare l'ambiente LXDE. Sarai riportato nuovamente all'interfaccia PIXEL con la schermata di login.

Ottenere la ISO

Avrai bisogno del file ISO di Debian con PIXEL. Puoi ricavare il file ISO dal DVD-ROM, utilizzando un programma chiamato InfraRecorder su Windows (**magpi.cc/2fP6wZT**), Utility Disco in Mac OS X, oppure utilizzando il comando **dd** in Linux. Tuttavia, è meglio scaricare l'ultima versione della ISO dal sito web della Fondazione Raspberry Pi (**magpi.cc/1MYYTMo**).

Usare Etcher

Ci sono molti modi per trasformare il file ISO in un unità avviabile. Secondo noi il più facile da usare è il software Etcher (**Etcher.io**). Abbiamo esaminato Etcher approfonditamente nel numero 50 di *The MagPi* (**magpi.cc/Back-issues**).

Usiamo generalmente Etcher per creare schede SD per avviare il nostro Raspberry Pi, ma si scopre essere altrettanto pratico per la masterizzazione di unità USB avviabili.

Apri Etcher e fai clic su Select Image. Seleziona il file ISO che contiene il sistema operativo Debian. Ora inserisci la chiave USB. Dovrebbe apparire in automatico sotto Select Drive (in caso contrario fai clic su Select Drive e scegli il disco corretto). Clicca Flash! per copiare il file ISO sul drive USB. Immetti la password, se richiesto, e fai clic su OK.

Avviare

Ora riavvia il computer con inserita la chiavetta USB. Se hai invece utilizzato il DVD, non ti

dimenticare di rimuoverlo dal lettore, altrimenti farai il boot da esso invece che dalla chiavetta USB.

Molti PC sono preimpostati per avviare da USB prima del disco fisso primario, ma se il tuo non lo è, dovresti accedere alle impostazioni del BIOS per modificare l'ordine di avvio (premi **F1** o segui le istruzioni su schermo durante l'avvio).

Crea disco persistente

Se hai installato Debian con pixel utilizzando un file ISO che hai creato dal DVD, è necessario trasformare lo spazio rimanente sul drive USB in una partizione persistente. Non c'è bisogno di farlo, invece, se hai installato Debian con pixel utilizzando la ISO scaricata dal sito della Fondazione Raspberry Pi Foundation (che include uno script che crea automaticamente la partizione di persistenza).

Apri una finestra del terminale e digita: **sudo apt installare gparted**. Quando ha finito, digitare **sudo GParted** per eseguire il programma con i privilegi di super user.

CHIAVETTA USB PER MAC

Se stai diventando matto per riuscire a far partire un Mac dalla la chiavetta USB, prova a formattare l'unità in due distinte partizioni. La prima partizione dovrà essere FAT32 e di dimensione di 1.5GB, con il nome boot. Il resto della chiavetta dovrà essere una partizione ext4 Linux con il nome persistence. Poi utilizza 7z per decomprimere il contenuto della ISO nella partizione FAT32. Abbiamo trovato queste utili istruzioni su CosmoLinux: magpi.cc/2ggdzrs

Fai clic sul menu Disks e scegli l'unità USB tra la lista dei tuoi dischi. Dovrebbe essere l'unità più piccola come dimensione..

Premi il pulsante destro del mouse sulla barra orizzontale grigia indicata come 'unallocated' e scegli New.

Inserisci **persistence** nel campo Label e fai clic su Add. Clicca l'icona verde Apply e premi Apply in Apply Operations to Device nella finestra di avviso. Fai click su Close quando vedi 'All operations successfully completed.'

Riavvia con **sudo reboot**. Questo monterà sia la volume di boot che il volume persistence.

Apri una finestra del terminale e digita **cd /media/pi**. Qui vedrai la partizione persistente montata (digita **ls -l** per visualizzare il volume ed i suoi permessi). Devi diventarne il proprietario per apportare modifiche. Inserisci **sudo chown pi persistence** per cambiare il proprietario del drive da root a pi. Digita **ls -l** per visualizzare di nuovo i permessi..

Usa **cd persistence/** per spostarti nella radice del drive. Ora digita **echo / union > persistence.conf** per creare il file di configurazione di persistenza. Usa **cat persistence.conf** per verificare il contenuto del file di configurazione. Dovrebbe essere solo '/union'.

Digita **sudo reboot** per l'ultima volta. Ora farai il boot in Debian con Pixel dal tuo drive USB.

Sopra Usa Etcher per registrare il file ISO in un USB drive Debian avviabile per il tuo PC

A sinistra Con la tua USB, potrai avviare direttamente in Debian PIXEL: è come usare un computer Raspberry Pi